Exercícios: Listas Encadeadas

1. Cite duas situações: uma onde seria mais vantajoso usar **vetores** e outra onde seria mais vantajoso usar **listas**.

2. Reescreva o código *nodes_creation.c* sem utilizar um vetor para armazenar os nós.

3. Como é determinado o **final** de uma lista encadeada?

4. Escreva uma rotina que receba uma lista encadeada de números inteiros como parâmetro e retorne a **quantidade de elementos** da lista indicada. Qual é a ordem de complexidade desta rotina?

5. Modifique as funções *push_back*() e *push_front*() dos códigos *intlist2.h* e *intlist2.c* de modo que retornem um número inteiro que indique ou o **sucesso** da operação ou o **código do erro** ocorrido. Capture e informe estes erros ao usuário modificando o código *testintlist2.c*.

6. Qual é a vantagem de se acrescentar o ponteiro *tail* numa lista encadeada? Qual é a desvantagem? Reescreva a rotina que insere um elemento **ao final** de uma lista encadeada que possua apenas o ponteiro *head*.

7. Se você estivesse desenvolvendo uma API que suportasse listas encadeadas em C, quais das 5 alternativas para o tratamento do caso especial **lista vazia** da remoção de um elemento do **início da lista** você usaria? Justifique sua resposta. Caso não opte por nenhuma das 5 alternativas, proponha e justifique uma nova solução.

8. Modifique o código *testintlist3.c* informando ao usuário qual foi o erro que ocorreu na remoção do elemento ao invés de usar a mensagem de erro padrão. *Sugestão:* capture o retorno das funções *pop_back*() e *pop_front*().

9. Quais seriam as vantagens e as desvantagens de se **desassociar** a recuperação de um elemento da lista com a sua remoção da mesma? Justifique suas escolhas.

10. Declara a estrutura dos nós e de uma lista encadeada de elementos do tipo *string*.

11. Qual seriam as vantagens e desvantagens de se criar uma lista **genérica**, onde os valores a serem armazenados seriam ponteiros do tipo **void** *?